A Generation Device of C Language Programming Design Based on Python

Xiaoming He[1], Haiqiang Geng[2], Lei Ding[1], Xiangrong Wang[1], Mengmeng Wu[1]
1.Shandong Vocational College of Information Technology, Weifang 261000, China.
2. State Grid Weifang Power Supply Company, Weifang 261000, China.

Abstract: In the design process of C language programs, it is generally necessary to follow the basic syntax of C language.And in some fields, the design process of C language programs also needs to follow basic industry standards, such as the MISRA C specification in the automotive electronics field.For handwritten code, it is difficult to keep each line of program written in compliance with these specifications.This paper presents a python-based C language generator.This device is implemented by python, through which some C language programs conforming to certain specifications can be implemented, such as variable definition, macro definition, structure type definition, enumeration type definition, etc.
Keywords: C Language Programming; Python; Specification

Introduction

The application field of C language is very extensive.In many ways, C language has great advantages as a programming language.However, in the process of writing C language programs, in addition to following the basic grammar of C language, for specific areas, it also needs to follow certain industry specifications.If a large amount of code is involved in a project, the process of writing C language programs is very slow.It also needs professional software to check whether it meets industry specifications after writing. If it does not, it needs to repeat the modification work many times.This is not conducive to fast iteration of software products.This paper describes a python-based C language generator.With this device, some C language program codes can be generated automatically, which can reduce the iteration cycle of software products and improve product quality.

1. Input File Description

Before using the C language generator, you need to fill in the table of the object that need to generate the code.This table is the input file.The table is named after the code module name.Different data objects can be managed using different worksheets.A table can contain the following worksheets:

1) Module Description:This worksheet contains information such as module owner and version number.

2) Measurement:This worksheet describes the type, variable name, storage location, and other information of the measurement quantity that needs to be defined.

3) Characteristic:This worksheet describes the type, variable name, storage location, and other information of the calibration quantity that needs to be defined.

4) Structure type:This worksheet describes the structure type definition.

5) Enumeration Type:This worksheet describes the enumeration type definition.

2. Operation Interface Design

To specify the location of the input file, use the “Tkinter” module in Python 3.11 to implement the interface design of the C language generator.The designed operation interface is shown in the following figure.When you click the button "click" in the interface, a file selection dialog box pops up.At this point, you can choose the input table file. When the selection is complete, the file path is displayed synchronously in the input box on the left.

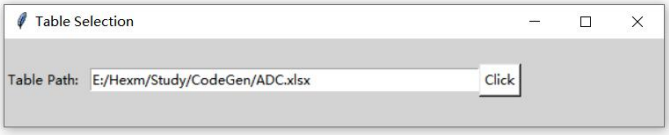

Figure 1 Operation interface

3. Table Information Extraction and Code Generation

3.1 “Module Description”worksheet Processing

The worksheet of “ModuleDesp” mainly completes the overall description of the module, including information such as the module owner and version number. This information is mainly used to generate the annotation section of C language files.Firstly, use the Pandas module in Python to read the contents of the worksheet "ModuleDesp", and then generate the extracted information into comments in the C language program.The content of the extracted "ModuleDesp" worksheet is shown in Figure 2. And the resulting C language notes are shown in Figure 3.

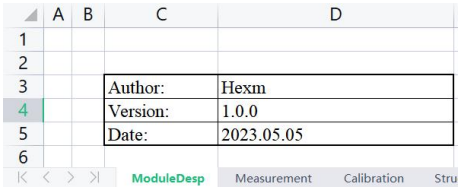

Figure 2 Module Description

```
/*************************************/
/*************************************/
/*************************************/
/*** Module Name: ADC ****************/
/*** Author : Hexm *******************/
/*** Version : 1.0.0 *****************/
/*** Modification date : 2023.05.05 ***/
/*** Revision history :      **********/
/*** 2023.05.05 Create File by Hexm****/
/*************************************/
/*************************************/
/*************************************/
```

Figure 3 Annotation

3.2 “Measurement”worksheet Processing

The measurement worksheet is filled with information such as variable names, variable types, and variable storage locations that need to be defined.Taking the automotive electronic control unit as an example, due to the limited internal storage space of the control unit, it is necessary to divide the storage area for the measurement quantity.So before generating the definition code for measurement quantities, it is necessary to first put the measurement quantities from the same storage area together.In this article, the dictionary in Python is used for storage. Using variable partitions as keys and variable types and names as values allows for centralized processing of variables in the same partition.The Python code for extracting measurement quantities into the dictionary MeaDict is as follows:

```
def read_Measurement():
global Filepath
global MeaDict
datas = pd.read_excel(Filepath, sheet_name='Measurement', header=0, usecols=[0,1,2,3])
for i in datas.index.values:
partition = datas.values[i, 3]
Type = datas.values[i, 1]
var = datas.values[i, 2]
MeaDict.setdefault(partition, []).append(Type)
MeaDict.setdefault(partition, []).append(var)
```

The content of the measurement worksheet is shown in Figure 4, and the content extracted from the MeaDict dictionary is shown in Figure 5.

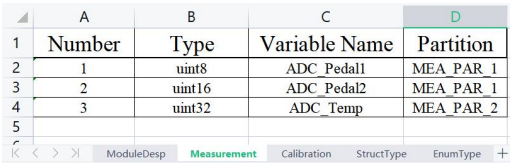

Figure 4 Measurement

```
MeaDict:
{
'MEA_PAR_1':['uint8','ADC_Pedal1','uint16','ADC_Pedal2'],
'MEA_PAR_2':['uint32','ADC_Temp']
}
```

Figure 5 Measurement Dictionary

After extracting the measurement into the “MeaDict” dictionary, the variable definition can be printed to the C language source file by traversing the dictionary. Pay attention to partition information when printing.The Python code for printing measurement quantity definitions is as follows, only the key parts of the code are listed here.By using two nested for loops, it is possible to quickly generate measurement quantity definition code.No matter how many variables are filled in the table, the following program can be used to quickly generate variable definitions in C language programs, which greatly improves the efficiency and quality of code writing.The definition of measurement quantity is generated into a. c file, and the declaration is synchronously generated into a. h file.

```
OutputFile = ModuleName + '.c'
OutputFile2 = ModuleName + '.h'
file = open(OutputFile,mode = 'a')
file2 = open(OutputFile2,mode = 'w')
for i in range(0,len(MeaDict),1):
key = list(MeaDict.keys())[i]
file.write("#define " + ModuleName + "_START_" + key + "\n")
file.write("#include \"Memmap.h\"\n")
file2.write("#define " + ModuleName + "_START_" + key + "\n")
file2.write("#include \"Memmap.h\"\n")
for j in range(0,len(MeaDict[key]),2):
file.write(MeaDict[key][j]+" "+MeaDict[key][j+1]+ ";\n")
file2.write("extern " + MeaDict[key][j]+" "+MeaDict[key][j+1]+ ";\n")
file.write("#define " + ModuleName + "_STOP_" + key + "\n")
file.write("#include \"Memmap.h\"\n\n")
file2.write("#define " + ModuleName + "_STOP_" + key + "\n")
file2.write("#include \"Memmap.h\"\n\n")
```

The definition and declaration of the generated measurement quantity are shown in Figure 6 and Figure For other worksheets, the processing method is similar to that of the "Measurement" worksheet.

```
/************************************/
/*********Variable Definition**********/
/************************************/
#define ADC_START_MEA_PAR_1
#include "Memmap.h"
uint8 ADC_Pedal1;
uint16 ADC_Pedal2;
#define ADC_STOP_MEA_PAR_1
#include "Memmap.h"

#define ADC_START_MEA_PAR_2
#include "Memmap.h"
uint32 ADC_Temp;
#define ADC_STOP_MEA_PAR_2
#include "Memmap.h"
```

Figure 6 Measurement Definition

```
/************************************/
/*********Extern Declaration**********/
/************************************/
#define ADC_START_MEA_PAR_1
#include "Memmap.h"
extern uint8 ADC_Pedal1;
extern uint16 ADC_Pedal2;
#define ADC_STOP_MEA_PAR_1
#include "Memmap.h"

#define ADC_START_MEA_PAR_2
#include "Memmap.h"
extern uint32 ADC_Temp;
#define ADC_STOP_MEA_PAR_2
#include "Memmap.h"
```

Figure 7 Measurement Declaration

4. Conclusion

In this paper, the objects that need to be defined in the C language program are filled in the form according to the classification. Using Python language as a tool, the information filled in the form is extracted, and finally the C language program code that conforms to the grammar and relevant specifications of the automotive electronics industry is generated.The device that does this is called a Python-based C language generator. This device uses tables to manage data

objects in C language program, which is clearer and easier to maintain. And it can generate standardized C language program code in batches, which is of great significance to achieve rapid iteration of large-scale engineering projects.

References

[1] Kondratyev D. A. & Nepomniaschy V. A..(2022).Automation of C Program Deductive Verification without Using Loop Invariants. Programming and Computer Software(5).

[2] Hu MZ & Zhang Y.(2022).An empirical study of the Python/C API on evolution and bug patterns. Journal of Software: Evolution and Process(2).

[3] Dai P, Wang YW, Jin DH, Gong YZ & Yang WJ.(2022). An improving approach to analyzing change impact of C programs. Computer Communications.

[4] Lennart Beringer & Andrew W. Appel.(2021).Abstraction and subsumption in modular verification of C programs. Formal Methods in System Design(prepublish). [5] Jim Pivarski, Peter Elmer & David Lange.(2020).Awkward Arrays in Python, C++, and Numba. EPJ Web of Conferences.